WHAT IS CLAIMED IS:

1. An interpreter that executes a program written in a programming language in cooperation with a processor, the interpreter comprising:

a module that calls a native code; and

a native code emulator that executes the native code through hardware emulation.

2. An interpreter according to claim 1, wherein the native code emulator includes a monitoring module to monitor a memory access instruction by the native code.

3. An interpreter according to claim 1, further comprising a table for memory regions that are managed by the interpreter wherein the table records information as to whether or not each of the memory regions is accessible from the native code.

4. An interpreter according to claim 3, wherein the table records information as to whether or not each of the memory regions is readable, writable or executable from the native code.

5. An interpreter according to claim 3, wherein, when the native

code emulator executes the native code, the monitoring module refers to the table to detect an illegal reference that is made when the memory access instruction is executed.

6. An interpreter according to claim 1, further comprising a determination module that makes a determination as to whether a target code in a program to be executed is an interpreter code or a native code, wherein, when the target code is determined to be a native code, the native code emulator processes the native code.

7. An interpreter according to claim 6, wherein, when transition between execution of an interpreter code and execution of a native code is performed by a native method call, the determination module does not make the determination until a native method call occurs.

8. An interpreter according to claim 1, wherein the native code emulator stores execution state of portion of the native code.

9. An interpreter according to claim 8, wherein internal state of the interpreter and the execution state of the portion of the native code are saved when the program is stopped and execution state of the program is saved.

10. An interpreter according to claim 9, wherein the execution state of the program saved is read out to restart execution of the program from a point where the program is stopped.

11. An interpreter that executes a programming language in cooperation with a processor, the interpreter comprising:

a module that calls a native code; and

a monitoring module that monitors a memory access instruction by the native code.

12. An interpreter according to claim 11, further comprising a table for memory regions that are managed by the interpreter wherein the table records information as to whether or not each of the memory regions is accessible from the native code.

13. An interpreter according to claim 12, wherein the table records information as to whether or not each of the memory regions is readable, writable or executable from the native code.

14. An interpreter according to claim 11, further comprising a native code emulator that executes the native code through hardware emulation.

15. An interpreter according to claim 14, wherein, when the native code emulator executes the native code, the monitoring module refers to the table to detect an illegal reference that is made when the memory access instruction is executed.

16. A native code execution method for an interpreter that has a native code calling function and executes a programming language in cooperation with a processor, the native code execution method comprising the steps of:

calling a native code; and

executing the native code by a native code emulator through hardware emulation.

17. A native code execution method according to claim 16, wherein the native code is not directly executed by hardware.

18. A native code execution method according to claim 17, further comprising the step of monitoring a memory access instruction by the native code.

19. A native code execution method according to claim 18, further comprising the steps of creating a table of memory regions that are managed by the interpreter, and recording in the table information as to

whether or not each of the memory regions is accessible from the native code.

20. A native code execution method according to claim 19, wherein the table records information as to whether or not each of the memory regions is readable, writable or executable from the native code.

21. A native code execution method according to claim 20, further comprising the step of, when the native code emulator executes the native code, referring to the table to detect an illegal reference that is made when the memory access instruction is executed.